



RECEIVED

OCT 21 2004

Technology Center 2100

Attorney Docket No. 5246 P 003  
(207301)

PATENT

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Application of: )  
SEILER et al. )  
Application No. 09/919,594 )  
Filing Date: July 31, 2004 ) Examiner: Dodds, Harold E.  
For: "Method and System for Information ) Group Art Unit: 2177  
Communication Between Positionees and )  
Positionors" )  
)

**DECLARATION OF PRIOR INVENTION IN THE UNITED STATES  
TO OVERCOME CITED PUBLICATION PURSUANT TO 37 C.F.R. 1.131**

BOX Fee Amendment  
Commissioner for Patents  
WASHINGTON, DC 20231

DEAR COMMISSIONER:

As a below named inventor, I hereby declare that:

1. This declaration is to establish conception and/or reduction to practice of the invention in this application in the United States prior to the earliest effective date of four cited Publications from the First Office Action dated March 19, 2004 and used as grounds of rejection. The four Publications include:

- 1) U.S. Patent Publication No. 2002/0059201 Effective Date: May 9, 2000
- 2) U.S. Patent Publication No. 2002/0059228 Effective Date: July 31, 2000
- 3) U.S. Patent No. 6,658,400 Effective Date: December 4, 2000
- 4) U.S. Patent No. 6,606,744 Effective Date: November 22, 1999

2. This declaration is submitted prior to a final rejection, and as such is timely. Accordingly, Applicants respectfully request the Examiner to enter this evidence into the record.

## **FACTS AND DOCUMENTARY EVIDENCE**

3. To establish a date of conception and/or reduction to practice of this application prior to the earliest effective date of the cited Publications, specifically, November 22, 1999, the attached Exhibit A is submitted as evidence. I respectfully note that this document is not presently being relied upon for establishing the earliest date of conception and/or reduction to practice of the invention of one or more pending claims, but is solely being used to establish that the date of conception and/or reduction to practice of the invention was prior to November 22, 1999.

3. At least as early as November 22, 1999, I conceived of the invention as originally claimed in the above-identified application. I worked diligently to reduce to practice the invention until filing of Provisional Patent Application 60/222,689 on August 2, 2000, and/or until actual reduction to practice of the invention.

4. I have reviewed the document comprising Exhibit A and find that this document represents what I believe to be a true copy of the original record generated contemporaneously with my research and development of the invention coinciding with the claims of the above-identified application. This document supports that there was a conception and/or reduction to practice of the invention of the claims in the above-identified application of a date earlier than November 22, 1999, the effective date of the relevant subject matter of the earliest cited Publication listed above.

5. All of the above events were performed in the United States of America.

## **DECLARATION**

6. AS A PERSON SIGNING BELOW:

I hereby certify that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code, and that such willful false statements may jeopardize the validity of the application or any patent issued thereon.

Attorney Docket No.: 5246 P 003

Application No.: 09/919,594

Filing Date July 31, 2004

For: Method and System for Information Communication Between Positionees and Positioners

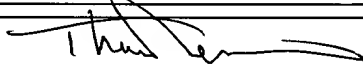
Full Name of 1st Joint Inventor:	<b>Margaret Seiler</b>
Residential Street Address:	<b>818 Sharon Drive</b>
City and State/Province:	<b>Dekalb, Illinois</b>
Country and Zip/Postal Code:	<b>U.S.A. 60115</b>
Citizenship:	<b>U.S.A.</b>
Mailing Address:	<b>Same as above.</b>
Inventor's Signature:	
Date:	

Attorney Docket No.: 5246 P 003

Application No.: 09/919,594

Filing Date July 31, 2004

For: Method and System for Information Communication Between Positionees and Positioners

Full Name of 2nd Joint Inventor:	<b>Thomas Revane</b>
Residential Street Address:	<del>435 Berkeley</del> 207 Fellows Court
City and State/Province:	<b>Elmhurst, Illinois</b>
Country and Zip/Postal Code:	<b>U.S.A. 60126</b>
Citizenship:	<b>U.S.A.</b>
Mailing Address:	<b>Same as above.</b>
Inventor's Signature:	
Date:	9-13-2004

Attorney Docket No.: 5246 P 003

Application No.: 09/919,594

Filing Date July 31, 2004

For: Method and System for Information Communication Between Positionees and Positioners

Full Name of 3rd Joint Inventor:	<b>Daniel Ambrecht</b>
Residential Street Address:	<b>14019 S. Kilpatrick</b>
City and State/Province:	<b>Crestwood, Illinois</b>
Country and Zip/Postal Code:	<b>U.S.A. 60445</b>
Citizenship:	<b>U.S.A.</b>
Mailing Address:	<b>Same as above.</b>
Inventor's Signature:	
Date:	

Attorney Docket No.: 5246 P 003

Application No.: 09/919,594

Filing Date July 31, 2004

For: Method and System for Information Communication Between Positionees and Positioners

Full Name of 4th Joint Inventor:	<b>William Backs</b>
Residential Street Address:	<b>2730 Simpson Street</b>
City and State/Province:	<b>Evanston, Illinois</b>
Country and Zip/Postal Code:	<b>U.S.A. 60201</b>
Citizenship:	<b>U.S.A.</b>
Mailing Address:	<b>Same as above.</b>
Inventor's Signature:	
Date:	

Attorney Docket No.: 5246 P 003

Application No.: 09/919,594

Filing Date July 31, 2004

For: Method and System for Information Communication Between Positionees and Positioners

Full Name of 5th Joint Inventor:	<b>William Bailey</b>
Residential Street Address:	<b>3751 N. Halsted Street #220</b>
City and State/Province:	<b>Chicago, Illinois</b>
Country and Zip/Postal Code:	<b>U.S.A. 60613</b>
Citizenship:	<b>U.S.A.</b>
Mailing Address:	<b>Same as above.</b>
Inventor's Signature:	
Date:	

Attorney Docket No.: 5246 P 003

Application No.: 09/919,594

Filing Date July 31, 2004

For: Method and System for Information Communication Between Positionees and Positioners

Full Name of 6th Joint Inventor:	<b>Andrew Bennett</b>
Residential Street Address:	<b>617 W. Dorset Drive</b>
City and State/Province:	<b>Wheaton, Illinois</b>
Country and Zip/Postal Code:	<b>U.S.A. 60187</b>
Citizenship:	<b>U.S.A.</b>
Mailing Address:	<b>Same as above.</b>
Inventor's Signature:	
Date:	



Attorney Docket No.: 5246 P 003

Application No.: 09/919,594

Filing Date July 31, 2004

For: Method and System for Information Communication Between Positionees and Positioners

Full Name of 7th Joint Inventor:	<b>Larry Books</b>
Residential Street Address:	<b>1104 Airport</b>
City and State/Province:	<b>Mt. Vernon, Illinois</b>
Country and Zip/Postal Code:	<b>U.S.A. 60115</b>
Citizenship:	<b>U.S.A.</b>
Mailing Address:	<b>Same as above.</b>
Inventor's Signature:	
Date:	

Attorney Docket No.: 5246 P 003

Application No.: 09/919,594

Filing Date July 31, 2004

For: Method and System for Information Communication Between Positionees and Positioners

Full Name of 8th Joint Inventor:	<b>James Bremenkamp</b>
Residential Street Address:	<b>944 W. Grace</b>
City and State/Province:	<b>Chicago, Illinois</b>
Country and Zip/Postal Code:	<b>U.S.A. 60614</b>
Citizenship:	<b>U.S.A.</b>
Mailing Address:	<b>Same as above.</b>
Inventor's Signature:	
Date:	

Attorney Docket No.: 5246 P 003

Application No.: 09/919,594

Filing Date July 31, 2004

For: Method and System for Information Communication Between Positionees and Positioners

Full Name of 9th Joint Inventor:	<b>Brad Brown</b>
Residential Street Address:	<b>230 E. Ontario, Unit 1202</b>
City and State/Province:	<b>Chicago, Illinois</b>
Country and Zip/Postal Code:	<b>U.S.A. 60611</b>
Citizenship:	<b>U.S.A.</b>
Mailing Address:	<b>Same as above.</b>
Inventor's Signature:	<i>Brad T. Brown</i>
Date:	<i>9-15-2004</i>

Attorney Docket No.: 5246 P 003

Application No.: 09/919,594

Filing Date July 31, 2004

For: Method and System for Information Communication Between Positionees and Positioners

Full Name of 10th Joint Inventor:	<b>Rodger Campbell</b>
Residential Street Address:	<b>4240 N. Kenmore 3S</b>
City and State/Province:	<b>Chicago, Illinois</b>
Country and Zip/Postal Code:	<b>U.S.A. 60613</b>
Citizenship:	<b>U.S.A.</b>
Mailing Address:	<b>Same as above.</b>
Inventor's Signature:	<i>Rodger Campbell</i>
Date:	<i>9-14-03</i>

Attorney Docket No.: 5246 P 003

Application No.: 09/919,594

Filing Date July 31, 2004

For: Method and System for Information Communication Between Positionees and Positioners

Full Name of 11th Joint Inventor:	<b>Ten Chu</b>
Residential Street Address:	<b>1578 Burning Bush Lane</b>
City and State/Province:	<b>Hoffman Estates, Illinois</b>
Country and Zip/Postal Code:	<b>U.S.A. 60195</b>
Citizenship:	<b>U.S.A.</b>
Mailing Address:	<b>Same as above.</b>
Inventor's Signature:	
Date:	

Attorney Docket No.: 5246 P 003

Application No.: 09/919,594

Filing Date July 31, 2004

For: Method and System for Information Communication Between Positionees and Positioners

Full Name of 12th Joint Inventor:	Michael Cooney
Residential Street Address:	733 Huntington Drive
City and State/Province:	Lake Zurich, Illinois
Country and Zip/Postal Code:	U.S.A. 60047
Citizenship:	U.S.A.
Mailing Address:	Same as above.
Inventor's Signature:	<i>Michael Cooney</i>
Date:	9/14/04

Attorney Docket No.: 5246 P 003

Application No.: 09/919,594

Filing Date July 31, 2004

For: Method and System for Information Communication Between Positionees and Positioners

Full Name of 13th Joint Inventor:	<b>Mitchell Daniels</b>
Residential Street Address:	<b>10701 Cass Road</b>
City and State/Province:	<b>Buffalo, Illinois</b>
Country and Zip/Postal Code:	<b>U.S.A. 62515</b>
Citizenship:	<b>U.S.A.</b>
Mailing Address:	<b>Same as above.</b>
Inventor's Signature:	
Date:	

Attorney Docket No.: 5246 P 003

Application No.: 09/919,594

Filing Date July 31, 2004

For: Method and System for Information Communication Between Positionees and Positioners

Full Name of 14th Joint Inventor:	<b>Rory Dickinson</b>
Residential Street Address:	<b>4238-102 Bloomington</b>
City and State/Province:	<b>Arlington Heights, Illinois</b>
Country and Zip/Postal Code:	<b>U.S.A. 60004</b>
Citizenship:	<b>U.S.A.</b>
Mailing Address:	<b>Same as above.</b>
Inventor's Signature:	
Date:	



Attorney Docket No.: 5246 P 003

Application No.: 09/919,594

Filing Date July 31, 2004

For: Method and System for Information Communication Between Positionees and Positioners

Full Name of 15th Joint Inventor:	<b>Martin Donnelly</b>
Residential Street Address:	<b>392 Cottage Hill</b>
City and State/Province:	<b>Elmhurst, Illinois</b>
Country and Zip/Postal Code:	<b>U.S.A. 60126</b>
Citizenship:	<b>U.S.A.</b>
Mailing Address:	<b>Same as above.</b>
Inventor's Signature:	
Date:	

Attorney Docket No.: 5246 P 003

Application No.: 09/919,594

Filing Date July 31, 2004

For: Method and System for Information Communication Between Positionees and Positioners

Full Name of 16th Joint Inventor:	<b>Ann Fish</b>
Residential Street Address:	<b>2717 N. Kenmore</b>
City and State/Province:	<b>Chicago, Illinois</b>
Country and Zip/Postal Code:	<b>U.S.A. 60614</b>
Citizenship:	<b>U.S.A.</b>
Mailing Address:	<b>Same as above.</b>
Inventor's Signature:	
Date:	

Attorney Docket No.: 5246 P 003

Application No.: 09/919,594

Filing Date July 31, 2004

For: Method and System for Information Communication Between Positionees and Positioners

Full Name of 17th Joint Inventor:	<b>Sandra Gates</b>
Residential Street Address:	<b>1712 7<sup>th</sup> Avenue</b>
City and State/Province:	<b>Maywood, Illinois</b>
Country and Zip/Postal Code:	<b>U.S.A. 60153</b>
Citizenship:	<b>U.S.A.</b>
Mailing Address:	<b>Same as above.</b>
Inventor's Signature:	
Date:	

Attorney Docket No.: 5246 P 003

Application No.: 09/919,594

Filing Date July 31, 2004

For: Method and System for Information Communication Between Positionees and Positioners

Full Name of 18th Joint Inventor:	<b>Sandra Grepling</b>
Residential Street Address:	<b>1 S. 269 Ardmore</b>
City and State/Province:	<b>Villa Park, Illinois</b>
Country and Zip/Postal Code:	<b>U.S.A. 60181</b>
Citizenship:	<b>U.S.A.</b>
Mailing Address:	<b>Same as above.</b>
Inventor's Signature:	
Date:	

Attorney Docket No.: 5246 P 003

Application No.: 09/919,594

Filing Date July 31, 2004

For: Method and System for Information Communication Between Positionees and Positioners

Full Name of 19th Joint Inventor:	<b>Michael Gulino</b>
Residential Street Address:	<b>9636 S. 46<sup>th</sup> Avenue</b>
City and State/Province:	<b>Oak Lawn, Illinois</b>
Country and Zip/Postal Code:	<b>U.S.A. 60453</b>
Citizenship:	<b>U.S.A.</b>
Mailing Address:	<b>Same as above.</b>
Inventor's Signature:	
Date:	

Attorney Docket No.: 5246 P 003  
Application No.: 09/919,594  
Filing Date July 31, 2004

For: Method and System for Information Communication Between Positionees and Positioners

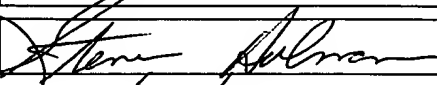
Full Name of 20th Joint Inventor:	<b>William Harrison</b>
Residential Street Address:	<b>3700 Tyler Street</b>
City and State/Province:	<b>Gary, Indiana</b>
Country and Zip/Postal Code:	<b>U.S.A. 46409</b>
Citizenship:	<b>U.S.A.</b>
Mailing Address:	<b>Same as above.</b>
Inventor's Signature:	
Date:	

Attorney Docket No.: 5246 P 003

Application No.: 09/919,594

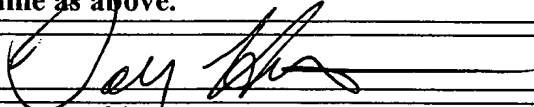
Filing Date July 31, 2004

For: Method and System for Information Communication Between Positionees and Positioners

Full Name of 21st Joint Inventor:	Steven Holman
Residential Street Address:	<del>8643 S. Hermitage</del> 1209 E. 168TH ST
City and State/Province:	<del>Chicago, Illinois</del> South Holland, IL
Country and Zip/Postal Code:	U.S.A. <del>60620</del> 60473
Citizenship:	U.S.A.
Mailing Address:	Same as above.
Inventor's Signature:	
Date:	09/13/04

Attorney Docket No.: 5246 P 003  
Application No.: 09/919,594  
Filing Date July 31, 2004

For: Method and System for Information Communication Between Positionees and Positioners

Full Name of 22nd Joint Inventor:	Douglas Horton
Residential Street Address:	2739 178 <sup>th</sup> Street
City and State/Province:	Lansing, Illinois
Country and Zip/Postal Code:	U.S.A. 60438
Citizenship:	U.S.A.
Mailing Address:	Same as above.
Inventor's Signature:	
Date:	9/16/04



Attorney Docket No.: 5246 P 003

Application No.: 09/919,594

Filing Date July 31, 2004

For: Method and System for Information Communication Between Positionees and Positioners

Full Name of 23rd Joint Inventor:	<b>Raymond Jackson</b>
Residential Street Address:	<b>1100 22<sup>nd</sup> Avenue</b>
City and State/Province:	<b>Bellwood, Illinois</b>
Country and Zip/Postal Code:	<b>U.S.A. 60104</b>
Citizenship:	<b>U.S.A.</b>
Mailing Address:	<b>Same as above.</b>
Inventor's Signature:	
Date:	

Attorney Docket No.: 5246 P 003

Application No.: 09/919,594

Filing Date July 31, 2004

For: Method and System for Information Communication Between Positionees and Positioners

Full Name of 24th Joint Inventor:	<b>Simon Jin</b>
Residential Street Address:	<b>3600 N. Lake Shore Drive #314</b>
City and State/Province:	<b>Chicago, Illinois</b>
Country and Zip/Postal Code:	<b>U.S.A. 60613</b>
Citizenship:	<b>U.S.A.</b>
Mailing Address:	<b>Same as above.</b>
Inventor's Signature:	
Date:	

Attorney Docket No.: 5246 P 003  
Application No.: 09/919,594  
Filing Date July 31, 2004

For: Method and System for Information Communication Between Positionees and Positioners

Full Name of 25th Joint Inventor:	<b>Christopher Jones</b>
Residential Street Address:	<b>7350 Carol Street, Apt. 2</b>
City and State/Province:	<b>Niles, Illinois</b>
Country and Zip/Postal Code:	<b>U.S.A. 60714</b>
Citizenship:	<b>U.S.A.</b>
Mailing Address:	<b>Same as above.</b>
Inventor's Signature:	
Date:	

Attorney Docket No.: 5246 P 003

Application No.: 09/919,594

Filing Date July 31, 2004

For: Method and System for Information Communication Between Positionees and Positioners

Full Name of 26th Joint Inventor:	<b>Gunter Kalogridis</b>
Residential Street Address:	<b>555 W. Madison #4812</b>
City and State/Province:	<b>Chicago, Illinois</b>
Country and Zip/Postal Code:	<b>U.S.A. 60661</b>
Citizenship:	<b>U.S.A.</b>
Mailing Address:	<b>Same as above.</b>
Inventor's Signature:	
Date:	

Attorney Docket No.: 5246 P 003

Application No.: 09/919,594

Filing Date July 31, 2004

For: Method and System for Information Communication Between Positionees and Positioners

Full Name of 27th Joint Inventor:	<b>Saptarshi Katwata</b>
Residential Street Address:	<b>25 NW Pt, Suite 600</b>
City and State/Province:	<b>Elk Grove Village, Illinois</b>
Country and Zip/Postal Code:	<b>U.S.A. 60007</b>
Citizenship:	<b>U.S.A.</b>
Mailing Address:	<b>Same as above.</b>
Inventor's Signature:	
Date:	

Attorney Docket No.: 5246 P 003

Application No.: 09/919,594

Filing Date July 31, 2004

For: Method and System for Information Communication Between Positionees and Positioners

Full Name of 28th Joint Inventor:	<b>Paul Kreiner</b>
Residential Street Address:	<b>1520 S. 7<sup>th</sup> Avenue</b>
City and State/Province:	<b>St. Charles, Illinois</b>
Country and Zip/Postal Code:	<b>U.S.A. 60174</b>
Citizenship:	<b>U.S.A.</b>
Mailing Address:	<b>Same as above.</b>
Inventor's Signature:	<i>Paul Kreiner</i>
Date:	<i>9/16/04</i>

Attorney Docket No.: 5246 P 003

Application No.: 09/919,594

Filing Date July 31, 2004

For: Method and System for Information Communication Between Positionees and Positioners

Full Name of 29th Joint Inventor:	<b>Breck Kuhnke</b>
Residential Street Address:	<b>1910 W. Greenleaf</b>
City and State/Province:	<b>Chicago, Illinois</b>
Country and Zip/Postal Code:	<b>U.S.A. 60626</b>
Citizenship:	<b>U.S.A.</b>
Mailing Address:	<b>Same as above.</b>
Inventor's Signature:	
Date:	

Attorney Docket No.: 5246 P 003  
Application No.: 09/919,594  
Filing Date July 31, 2004

For: Method and System for Information Communication Between Positionees and Positioners

Full Name of 30th Joint Inventor:	<b>Louis Lembcke</b>
Residential Street Address:	<b>344 S. 6<sup>th</sup> Avenue</b>
City and State/Province:	<b>La Grange, Illinois</b>
Country and Zip/Postal Code:	<b>U.S.A. 60525</b>
Citizenship:	<b>U.S.A.</b>
Mailing Address:	<b>Same as above.</b>
Inventor's Signature:	
Date:	



Attorney Docket No.: 5246 P 003

Application No.: 09/919,594

Filing Date July 31, 2004

For: Method and System for Information Communication Between Positionees and Positioners

Full Name of 31st Joint Inventor:	Timothy Lenning
Residential Street Address:	36 Potowattomie Court
City and State/Province:	Naperville, Illinois
Country and Zip/Postal Code:	U.S.A. 60563
Citizenship:	U.S.A.
Mailing Address:	Same as above.
Inventor's Signature:	<i>Timothy Lenning</i>
Date:	9/14/04

Attorney Docket No.: 5246 P 003

Application No.: 09/919,594

Filing Date July 31, 2004

For: Method and System for Information Communication Between Positionees and Positioners

Full Name of 32nd Joint Inventor:	<b>Thomas Lepore</b>
Residential Street Address:	<b>308 Walnut</b>
City and State/Province:	<b>Elmhurst, Illinois</b>
Country and Zip/Postal Code:	<b>U.S.A. 60126</b>
Citizenship:	<b>U.S.A.</b>
Mailing Address:	<b>Same as above.</b>
Inventor's Signature:	
Date:	

Attorney Docket No.: 5246 P 003

Application No.: 09/919,594

Filing Date July 31, 2004

For: Method and System for Information Communication Between Positionees and Positioners

Full Name of 33rd Joint Inventor:	Angel Li
Residential Street Address:	3254 N. Racine Avenue
City and State/Province:	Chicago, Illinois
Country and Zip/Postal Code:	U.S.A. 60657
Citizenship:	U.S.A.
Mailing Address:	Same as above.
Inventor's Signature:	
Date:	

Attorney Docket No.: 5246 P 003  
Application No.: 09/919,594  
Filing Date July 31, 2004

For: Method and System for Information Communication Between Positionees and Positioners

Full Name of 34th Joint Inventor:	<b>Gary Mayer</b>
Residential Street Address:	<b>1521 Countryside Drive</b>
City and State/Province:	<b>Buffalo Grove, Illinois</b>
Country and Zip/Postal Code:	<b>U.S.A. 60089</b>
Citizenship:	<b>U.S.A.</b>
Mailing Address:	<b>Same as above.</b>
Inventor's Signature:	
Date:	

Attorney Docket No.: 5246 P 003

Application No.: 09/919,594

Filing Date July 31, 2004

For: Method and System for Information Communication Between Positionees and Positioners

Full Name of 35th Joint Inventor:	<b>Joseph Mindock</b>
Residential Street Address:	<b>981 Lakeview Court</b>
City and State/Province:	<b>Oswego, Illinois</b>
Country and Zip/Postal Code:	<b>U.S.A. 60543</b>
Citizenship:	<b>U.S.A.</b>
Mailing Address:	<b>Same as above.</b>
Inventor's Signature:	
Date:	

Attorney Docket No.: 5246 P 003

Application No.: 09/919,594

Filing Date July 31, 2004

For: Method and System for Information Communication Between Positionees and Positioners

Full Name of 36th Joint Inventor:	<b>Karl Moulton</b>
Residential Street Address:	<b>P.O. Box 252</b>
City and State/Province:	<b>Peoria, Illinois</b>
Country and Zip/Postal Code:	<b>U.S.A. 61650</b>
Citizenship:	<b>U.S.A.</b>
Mailing Address:	<b>Same as above.</b>
Inventor's Signature:	
Date:	

Attorney Docket No.: 5246 P 003

Application No.: 09/919,594

Filing Date July 31, 2004

For: Method and System for Information Communication Between Positionees and Positioners

Full Name of 37th Joint Inventor:	<b>Brent Peebles</b>
Residential Street Address:	<b>1752 W. Cornelia</b>
City and State/Province:	<b>Chicago, Illinois</b>
Country and Zip/Postal Code:	<b>U.S.A. 60657</b>
Citizenship:	<b>U.S.A.</b>
Mailing Address:	<b>Same as above.</b>
Inventor's Signature:	
Date:	

Attorney Docket No.: 5246 P 003

Application No.: 09/919,594

Filing Date July 31, 2004

For: Method and System for Information Communication Between Positionees and Positioners

Full Name of 38th Joint Inventor:	<b>Denise Pike</b>
Residential Street Address:	<b>1712 Frost Lane</b>
City and State/Province:	<b>Naperville, Illinois</b>
Country and Zip/Postal Code:	<b>U.S.A. 60564</b>
Citizenship:	<b>U.S.A.</b>
Mailing Address:	<b>Same as above.</b>
Inventor's Signature:	
Date:	



Attorney Docket No.: 5246 P 003  
Application No.: 09/919,594  
Filing Date July 31, 2004

For: Method and System for Information Communication Between Positionees and Positioners

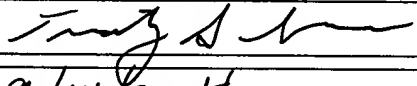
Full Name of 39th Joint Inventor:	<b>Andrzej Placzek</b>
Residential Street Address:	<b>8417 W. Bryn Mawr 3S</b>
City and State/Province:	<b>Chicago, Illinois</b>
Country and Zip/Postal Code:	<b>U.S.A. 60631</b>
Citizenship:	<b>U.S.A.</b>
Mailing Address:	<b>Same as above.</b>
Inventor's Signature:	
Date:	

Attorney Docket No.: 5246 P 003

Application No.: 09/919,594

Filing Date July 31, 2004

For: Method and System for Information Communication Between Positionees and Positioners

Full Name of 40th Joint Inventor:	<b>Timothy Saar</b>
Residential Street Address:	<b>2737 Hawthorne Street</b>
City and State/Province:	<b>Franklin Park, Illinois</b>
Country and Zip/Postal Code:	<b>U.S.A. 60131</b>
Citizenship:	<b>U.S.A.</b>
Mailing Address:	<b>Same as above.</b>
Inventor's Signature:	
Date:	<b>9/14/2004</b>

Attorney Docket No.: 5246 P 003

Application No.: 09/919,594

Filing Date July 31, 2004

For: Method and System for Information Communication Between Positionees and Positioners

Full Name of 41st Joint Inventor:	<b>Mark Schlosser</b>
Residential Street Address:	<b>226 N. Clinton, #209</b>
City and State/Province:	<b>Chicago, Illinois</b>
Country and Zip/Postal Code:	<b>U.S.A. 60661</b>
Citizenship:	<b>U.S.A.</b>
Mailing Address:	<b>Same as above.</b>
Inventor's Signature:	
Date:	

Attorney Docket No.: 5246 P 003

Application No.: 09/919,594

Filing Date July 31, 2004

For: Method and System for Information Communication Between Positionees and Positioners

Full Name of 42nd Joint Inventor:	<b>Susan Schumerth</b>
Residential Street Address:	<b>2645 N. Southport #2</b>
City and State/Province:	<b>Chicago, Illinois</b>
Country and Zip/Postal Code:	<b>U.S.A. 60614</b>
Citizenship:	<b>U.S.A.</b>
Mailing Address:	<b>Same as above.</b>
Inventor's Signature:	
Date:	

Attorney Docket No.: 5246 P 003

Application No.: 09/919,594

Filing Date July 31, 2004

For: Method and System for Information Communication Between Positionees and Positioners

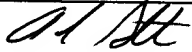
Full Name of 43rd Joint Inventor:	Scott Sellers
Residential Street Address:	2645 N. Southport #2
City and State/Province:	Chicago, Illinois
Country and Zip/Postal Code:	U.S.A. 60614
Citizenship:	U.S.A.
Mailing Address:	Same as above.
Inventor's Signature:	
Date:	

Attorney Docket No.: 5246 P 003

Application No.: 09/919,594

Filing Date July 31, 2004

For: Method and System for Information Communication Between Positionees and Positioners


Full Name of 44th Joint Inventor:	Andrew Sutter
Residential Street Address:	841 Jewett
City and State/Province:	Woodstock, Illinois
Country and Zip/Postal Code:	U.S.A. 60098
Citizenship:	U.S.A.
Mailing Address:	Same as above.
Inventor's Signature:	
Date:	9/15/04

Attorney Docket No.: 5246 P 003

Application No.: 09/919,594

Filing Date July 31, 2004

For: Method and System for Information Communication Between Positionees and Positioners

Full Name of 45th Joint Inventor:	Korol Taylor
Residential Street Address:	408 Asbury Avenue
City and State/Province:	Evanston, Illinois
Country and Zip/Postal Code:	U.S.A. 60202
Citizenship:	U.S.A.
Mailing Address:	Same as above.
Inventor's Signature:	
Date:	9/14/04

Attorney Docket No.: 5246 P 003

Application No.: 09/919,594

Filing Date July 31, 2004

For: Method and System for Information Communication Between Positionees and Positioners

Full Name of 46th Joint Inventor:	Eric Toennies
Residential Street Address:	1034 Colonial
City and State/Province:	Naperville, Illinois
Country and Zip/Postal Code:	U.S.A. 60540
Citizenship:	U.S.A.
Mailing Address:	Same as above.
Inventor's Signature:	<i>Eric Toennies</i>
Date:	<i>9/14/2004</i>



Attorney Docket No.: 5246 P 003

Application No.: 09/919,594

Filing Date July 31, 2004

For: Method and System for Information Communication Between Positionees and Positioners

Full Name of 47th Joint Inventor:	<b>William Vonderhaar</b>
Residential Street Address:	<b>2024 N. Racine</b>
City and State/Province:	<b>Chicago, Illinois</b>
Country and Zip/Postal Code:	<b>U.S.A. 60614</b>
Citizenship:	<b>U.S.A.</b>
Mailing Address:	<b>Same as above.</b>
Inventor's Signature:	
Date:	

Attorney Docket No.: 5246 P 003  
Application No.: 09/919,594  
Filing Date July 31, 2004

For: Method and System for Information Communication Between Positionees and Positioners

Full Name of 48th Joint Inventor:	<b>M. John Way</b>
Residential Street Address:	<b>1007 Sunset</b>
City and State/Province:	<b>Wheaton, Illinois</b>
Country and Zip/Postal Code:	<b>U.S.A. 60187</b>
Citizenship:	<b>U.S.A.</b>
Mailing Address:	<b>Same as above.</b>
Inventor's Signature:	
Date:	

Attorney Docket No.: 5246 P 003  
Application No.: 09/919,594  
Filing Date July 31, 2004

For: Method and System for Information Communication Between Positionees and Positioners

Full Name of 49th Joint Inventor:	<b>Lisa Winfrey</b>
Residential Street Address:	<b>411 Maude</b>
City and State/Province:	<b>Joliet, Illinois</b>
Country and Zip/Postal Code:	<b>U.S.A. 60433</b>
Citizenship:	<b>U.S.A.</b>
Mailing Address:	<b>Same as above.</b>
Inventor's Signature:	
Date:	

Attorney Docket No.: 5246 P 003

Application No.: 09/919,594

Filing Date July 31, 2004

For: Method and System for Information Communication Between Positionees and Positioners

Full Name of 50th Joint Inventor:	<b>Carol Yanowitz</b>
Residential Street Address:	<b>2132 N. Hoyne</b>
City and State/Province:	<b>Chicago, Illinois</b>
Country and Zip/Postal Code:	<b>U.S.A. 60647</b>
Citizenship:	<b>U.S.A.</b>
Mailing Address:	<b>Same as above.</b>
Inventor's Signature:	
Date:	

Attorney Docket No.: 5246 P 003  
Application No.: 09/919,594  
Filing Date July 31, 2004

For: Method and System for Information Communication Between Positionees and Positioners

Full Name of 51st Joint Inventor:	<b>Youan Zeng</b>
Residential Street Address:	<b>9052 Federal Ct. Apt. 1C</b>
City and State/Province:	<b>Des Plaines, Illinois</b>
Country and Zip/Postal Code:	<b>U.S.A. 60016</b>
Citizenship:	<b>U.S.A.</b>
Mailing Address:	<b>Same as above.</b>
Inventor's Signature:	
Date:	

Attorney Docket No.: 5246 P 003  
Application No.: 09/919,594  
Filing Date July 31, 2004


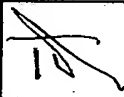
For: Method and System for Information Communication Between Positionees and Positioners

Full Name of 52nd Joint Inventor:	<b>Mark Zgarrrick</b>
Residential Street Address:	<b>1626 Madison St.</b>
City and State/Province:	<b>Evanston, Illinois</b>
Country and Zip/Postal Code:	<b>U.S.A. 60202</b>
Citizenship:	<b>U.S.A.</b>
Mailing Address:	<b>Same as above.</b>
Inventor's Signature:	
Date:	

A

Application Architecture Deliverable Approval Cover Sheet

Attached is the Application Architecture Deliverable for the Illinois Skills Match Project. It has been reviewed and approved as satisfying Chicago Systems Group's requirements for this deliverable as described in their proposal.

	Initials	Date
Sandy Grepling IDES ISB		9/27/99
Tom Revane IDES ISB		9/27/99
Mike Cooney Chicago Systems Group	MC	9/29/99



# Application Architecture

## Section Contents

1. Application Architecture Overview
2. Web Site Architecture
3. Online Components
4. Batch Components
5. Infrastructure Components

# **Section 1**

## **Application Architecture Overview**

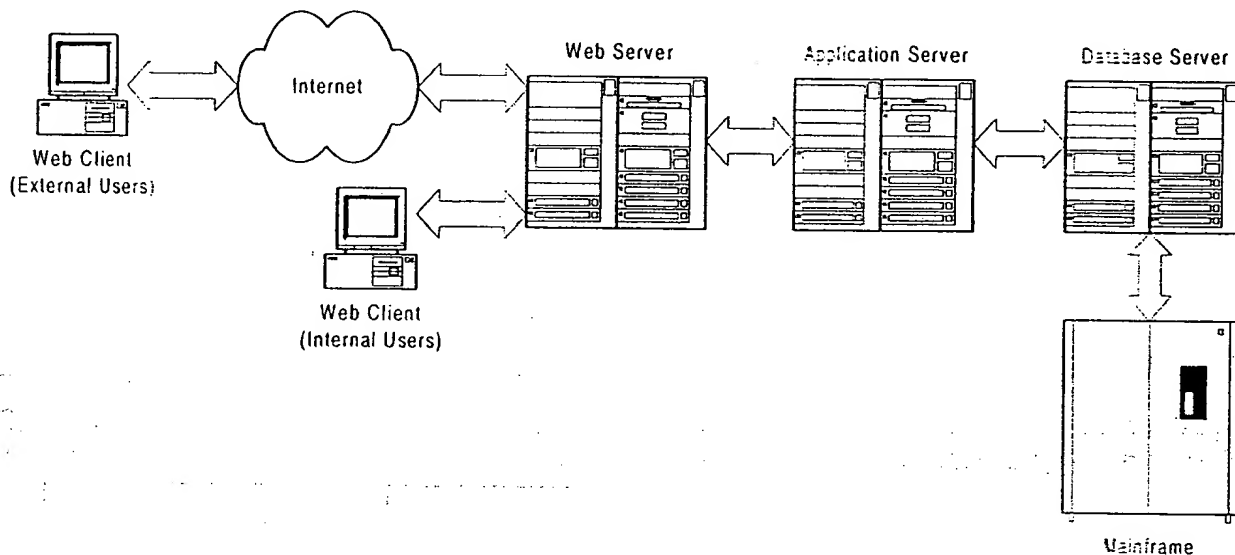
# Application Architecture Overview

This document describes the overall organization of the ISM system from the application developer's perspective. The environment in which the application is developed and executed is discussed, along with some high level design approaches, program elements, and specific application component design issues and details.

## Execution Environment

Figure 1-1 below diagrams the computer system components and their interaction at a logical level.

Figure 1-1 - Logical Architecture



The web clients access the ISM system by sending and receiving requests and results to a web server. All interactive screens are displayed by formatting an HTML page and delivering its content to the user's browser. The web server sends requests for dynamic content to a separate application server. This server accesses the database server to retrieve data, assembles an HTML response, and then delivers the page back to the web server. Batch interface programs execute on the database server to transfer data between the ISM database and existing mainframe applications.

## Application Components

The ISM application can be broken down into five basic components.

- Web site components
- Online application components
- Batch components
- Reporting components
- Infrastructure Components

### Web Site Components

The ISM system is accessed by a web browser. All user interface is handled through the web server by sending HTML to the client and responding to the client's HTTP requests. The web server also holds static content such as image files. In addition to the web server, the application server generates web content. The application server merges data from the database with HTML to generate the final HTML stream that gets delivered to the client browser. This operation is performed by a *Java Server Page (JSP)*. A JSP is a HTML page with special Java programming logic embedded in it.

## Application Components

The application logic of the ISM system is primarily implemented using Netscape Application Server (NAS). NAS *Servlets* implement the majority of the business logic. A servlet is a Java program that executes on the NAS server in the context of a user *session*. Every user of the ISM system will enter through a logon process. At the time of logon, the user session will be instantiated. From that point on, each HTTP request from that user that goes to the application servers will execute in the context set up when that user logged on.

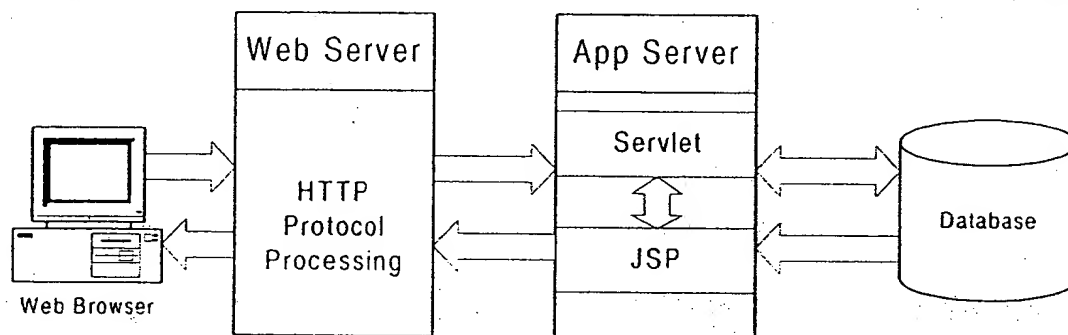
The Servlet accepts data from the web page where the data was entered. Data validation and database processing is then performed. The process continues with the next JSP being called to present the next page.

Another component of the application is stored procedures in the DB2 database server.

On the client side, some application logic and special user interface presentation mechanics are handled by JavaScript.

The flow of information between the web and application components is shown in figure 1-2.

Figure 1-2 - Data Flow Overview



## Batch Components

Some of the functions performed by the ISM system run at regular intervals and are scheduled to run in batch mode. These functions are primarily in the area of interfaces to existing mainframe systems. The programs run on the database server.

All batch jobs are Korn shell scripts. Inside the script is the execution of Korn shell commands, Perl Scripts, and COBOL programs, which often make use of stored procedures in the database.

## Reporting Components

Standard reports are available from the ISM system to support the various offices. These are typically monthly reports that will be delivered either electronically or manually depending on the capabilities of

the individual office. These reports are run in batch mode on the database server.

## Infrastructure Components

Functions that are outside of the business logic category, but that form a foundation for the inner workings of the system are classified as infrastructure components. These functions are responsible for such things as implementing the security system, error reporting and recovery, and other basic capabilities in the application that are shared amongst the other components. The infrastructure components for ISM are implemented through the base *object model* in Java, and *extension* modules that enhance the capabilities of NAS and the base operating system.

## Programming Tools and Environments

This section describes the programming languages and development tools used. Both the development and deployment environment are addressed.

### Web Page Development

The creation of static web pages and the HTML templates for use by NAS is performed on the developer's Windows NT workstation using the following products:

- MacroMedia Dreamweaver
- Allaire HomeSite
- NetObjects ScriptBuilder

Static web pages are deployed to the Netscape Enterprise Server (NES) web server. HTML to be used in JSPs are deployed to the NAS server.

### Application Logic Development

The application logic is comprised of the program modules that support the online web application and the batch functions.

#### Web Site Logic

The web site application logic is developed using Netscape Application Builder (NAB). NAB is used to create the JSPs and Servlets. Both of these components contain Java code to implement the application logic. Symantec Visual Café is used to develop the Java modules. It provides more features than NAB to make Java development more efficient.

Development using these products is performed on the Windows NT workstation. Deployment is to the NAS servers.

#### Extensions

NAS and other extensions are developed in Microsoft Visual C++. Deployment is on the NAS server. Once developed, the source code is moved to the NAS server and re-compiled and an installation is performed to connect the Extension to the NAS server.

#### Stored Procedures

IBM DB2 Universal Database Extended Enterprise Edition (UDB EEE) serves as the database repository for the ISM system. UDB EEE comes packaged with DB2 Stored Procedure Builder. This product assists in the development and testing of stored procedures. It can also be used to deploy the stored procedure to the database server. Deployment can also be done with a traditional "CREATE PROCEDURE" statement using an interactive SQL session with the database.

#### Batch Programs

The batch programs create datasets for upload to the mainframe, and process datasets created on the mainframe for processing into the database. These batch programs are Korn Shell scripts that execute Perl and COBOL programs. Perl is developed using a standard text editor. Merant Microfocus COBOL is the tool used to develop COBOL programs. Development is performed on the Windows NT workstation. Deployment is to the database server.

Web Site Architecture

Web Site Architecture

## Section 2

# Web Site Architecture



initial home page identifies the user type and requests a username and password. At this point, secure sockets layer (SSL) is used for transmitting this information to the web server. At this point, a number of evaluations are performed on the client browser. Once the browser capabilities and the user have been authenticated, an appropriate opening menu page is displayed depending on the user type.

### Menu Pages

Menu pages are displayed as appropriate for the type of user. A menu page is very straightforward. It contains no input controls, but just links to other pages in the system. Some conditional processing is performed to show or hide specific menu options based on the user's permissions. These decisions are made when the page is constructed on the application server.

### Search Pages

Search pages accept search criteria, and then execute a database search for data with matching criteria. After the search completes, a list page is built showing the results if one or more matching records is found. If no matching records are found, the search page is redisplayed with an error message.

### List Pages

These pages list several rows of information from the database. This is typically a result set from a database search. Each result record is a link that can be used to present the detail page for that data row. Optionally, each line in the list may also contain a checkbox that can be used to select a subset of records. The selected set of records can span multiple list pages.

Initially, the result set is divided up into pages. If the result set requires more than one page of list information, navigation buttons will be available to proceed to the next or previous page as necessary.

When a user selects a detail record, and then returns to the list view, the user will return to the same list page that contained the detail record most recently viewed.

Other activity in the ISM system may introduce or eliminate records from the user's result set. However, once the list is generated, it remains static until the user requests for the information to be refreshed. When necessary, some processes force a refresh to occur.

### Detail Pages

When complete detail on a record of information is requested, a detail page is presented. If a user requested the detail from a list page, then options on the detail page will exist to move through the list in detail view and an option to return to list view will be in place. If a subset of records was defined on the list page, then that subset defines the context of what the next/previous navigation will present to the user.

In simpler cases, detail pages are displayed from other non-list pages, or used for data entry purposes.

## General Web Page Issues

### Caching of Pages

In general, the dynamic nature of the ISM web pages makes it unlikely that the caching features of proxy servers and cache servers would be of any help to the performance of the system. In addition, cached pages may erroneously be sent to a client containing stale data. To disable caching, each page contains a pragma statement to turn caching off for proxies and cache servers.

Caching on the browser should be turned on. This enables certain functions to be performed from the client side using history to go back to previous pages.

## Page Organization and Flow

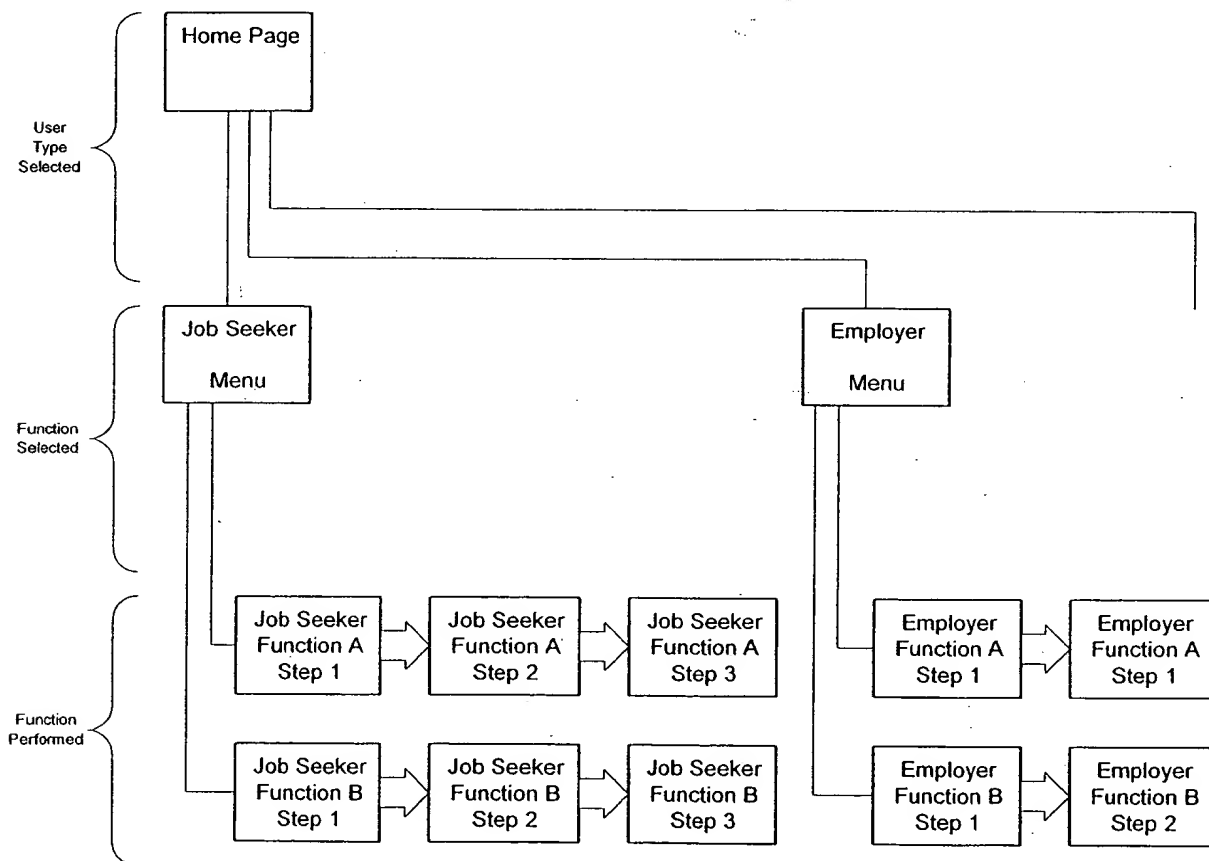
# Web Site Architecture

This section describes the web site issues and architecture of the ISM system. These issues include general web site organization, page layouts, page flow, interface mechanics, how the pages perform various tasks, and browser issues.

## General Organization

Figure 2-1 illustrates the general organization of web pages in the ISM web site.

Figure 2-1 - General Organization



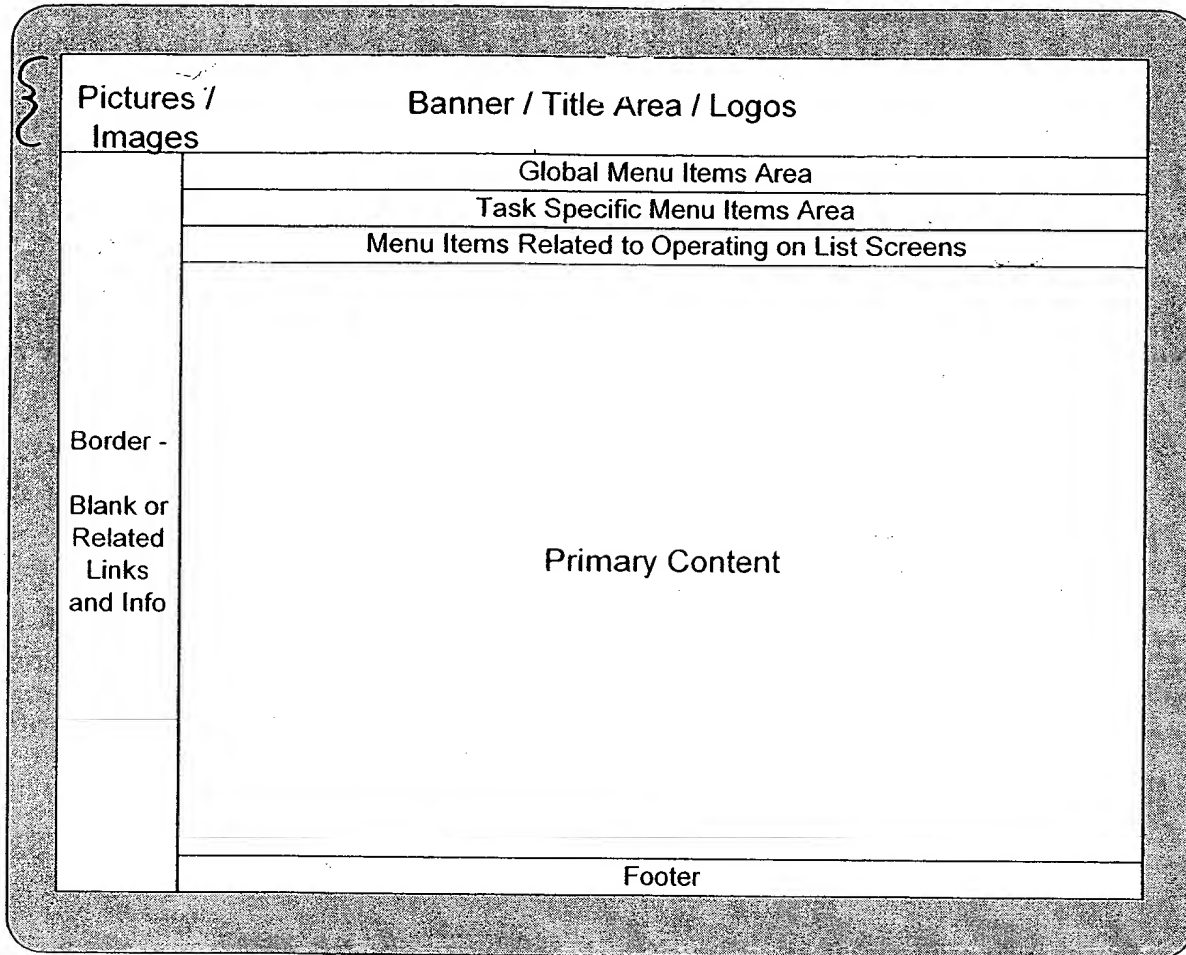
At the top level, there is a main page. On that page, the user makes a choice to indicate what type of user they are, and must enter a valid username and password, or go through a registration process. Categories of users are job seeker, employer, staff, and administrator. Once the user type is identified and their username and password is authenticated, a customized menu of function options is available on the screen. Once a function is selected, either a single page, or a series of pages will be presented to complete whatever process steps need to be performed.

## Look and Feel

### General Look and Feel

An extensive web site prototype has been developed for the ISM system. The prototype defines the specific layout of the page, and the look and feel of the web site. The position of controls, and graphical elements is used as a model for the final product. Figure 2-2 provides a general layout description of a standard web page.

Figure 2-2 - Standard Page Layout



The top banner portion provides a title and logos. The upper left box contains various graphics, such as a picture of the Governor, etc. A horizontal strip of global controls is always displayed below the top banner. When applicable, a horizontal strip of controls that are specific to the current page appears below the global control strip. If the page is a list page, a third strip of menu options is available directly below the task specific menu. The main body of the page with the primary content follows that. The body is where data elements and input/output is performed. A vertical strip of controls runs along the left hand side, providing an additional set of the global controls. Other links are provided there as well when appropriate to the user type and the function they are performing.

## Specific Page Types

The pages in the ISM web site can be broken down into five basic categories: the home/login page, menu pages, search pages, list pages, and detail pages.

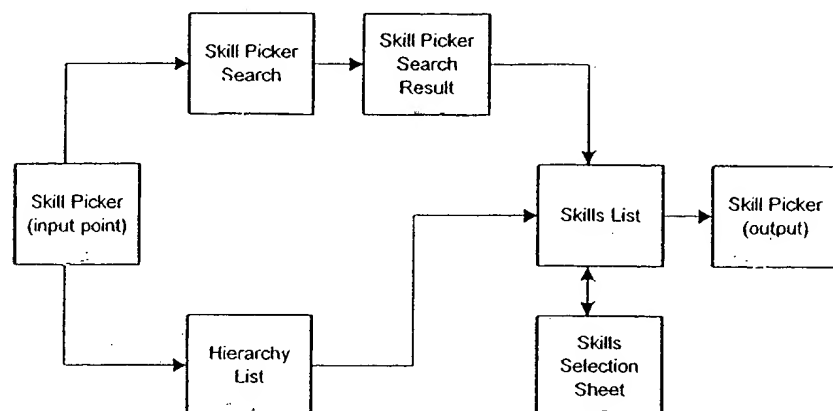
### Login Page

The home/login page is in its own category due to some rather complex processing requirements. The

### Figure 2-3 - Navigation Map



Figure 2-4 - Skill Picker Detail



## Global Navigation

Regardless of the user's location in the ISM system, there are global navigation options that allow the user to quickly access major functions, such as the user's main menu and logging off of the system.

## Logon, Sessions, and Security Issues

### Logging On

In order to implement security, and to support the NAS session architecture, all users of the ISM system must identify themselves to the system by logging on. Employers and job seekers go through a self-service registration process the first time they enter the system. Staff users are set up by ISM administrative staff. Staff users are assigned to various groups in order to implement the security architecture for the application.

### Session Setup

When the logon page is submitted, a session is created on NAS. In order for NAS to be able to identify what client session is making requests to it, a cookie holding the session ID is sent to the client. Subsequent requests from the client browser always include the cookie.

### Security Issues

Security is provided on the transmission of sensitive information through the use of secure sockets layer (SSL). HTTP browser transmissions using SSL, commonly referred to as HTTPS, protects the data contents through encryption. This technique is used for HTTP transmissions where either the request or response contain sensitive information.

## Browser Issues

### Required Features

In order to support the complex requirements of the ISM application, several advanced browser features are required. Some features are not available in older browsers. Most can be turned off by the user. The home page of the ISM system evaluates the user's browser for version and features. If the browser

is found to be lacking some of the requirements, the user is notified of what is required and how to gain access to those features. Refer to the ISM *Technical Architecture* document for full details on browser requirements.

### New Browser Window Launch

Once the user has logged in, or in any other way proceeds off of the home page (i.e. job seeker registration), a new browser window is opened. In the new window, the browser's tool bar and menu bar are disabled. This reduces the likelihood that the user would attempt to perform navigation that would upset the flow of transactions in the system and offers some additional screen space to use.

### Client Side Programming

Tools exist for writing program logic into the web page for execution at the browser. This is done in a very limited way using JavaScript on the pages of the ISM system. Simple checks, such as the existence of data in a required field and basic data format compliance make use of this technique. JavaScript is also used to implement some of the more complex user interface elements. Extensive data validation and business logic, however, is performed entirely on the application server. This provides for a more straightforward design and a focal point for maintenance and testing.

## **Section 3**

# **Online Components**

# Online Components

This section presents the structure of the online portion of the ISM application. General topics in this section include the mechanics of presenting the user interface, screen types, database access, input validation, error handling, and security. Issues related to specific sections of the application are discussed in detail, where important design decisions have been made.

## General Mechanics and Approaches

This section presents some rather broad topics and general approaches to implementing the online system.

### Screen Generation

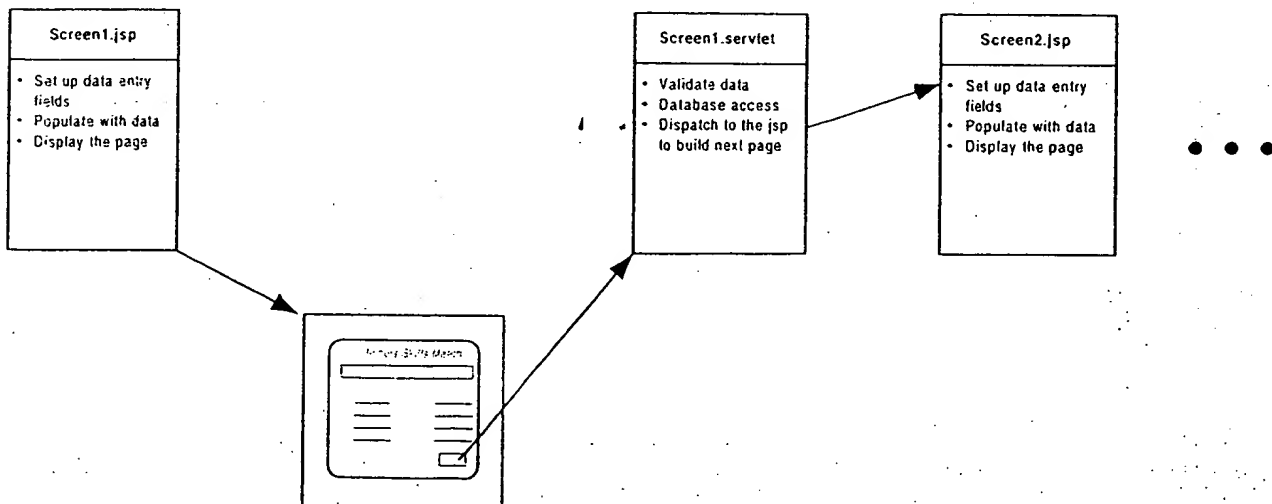
The mechanics of generating a screen begins with the user's browser request. These requests are always sent to one of the ISM web servers. If the request is for a static HTML page or other static content such as an image, the web server handles the request by itself. In the ISM system, the only static HTML page is the login page. The rest of the page requests are references to Servlets. In these cases, the requests are forwarded from the web server to the application server that is best suited to handle that request at that time. The best suited server is determined through load balancing information that flows between the application servers, and from the application servers to the web servers.

The normal processing of an online screen typically involves several steps:

1. Build the screen with input fields and any other controls
2. Process the input values, perform validation and database access.
3. Provide a response. Typically, this would be an error message or the presentation of the next screen in the process.

Programmatically, these functions are split into separate modules. The building of the screen is performed by a Java Server Page (JSP) for that screen. When the page is submitted for processing by the user, a Java Servlet is called. After processing the information, the Servlet chains to the next JSP. Figure 3-1 illustrates this process.

Figure 3-1 - Screen Generation





## Java Servlets

With the exception of the home and login pages, all pages in the ISM system are references to Servlets. The Servlet responds to the form submitted to it, and then dispatches a JSP to build the next screen.

## Java Server Pages (JSPs)

Java Server Pages are used to build the HTML response page back to the user after business logic processing has been completed by the Servlet. JSPs contain HTML and some Java code to retrieve information from the database, do security checking, etc.

To facilitate code re-use and a modular design approach, some JSP templates master templates have been created to serve as a starting point for all JSPs used in ISM. These templates contain some header and footer information as well as standard JavaScript functions needed by all web pages in the system.

## Database Access

The database management system (DBMS) for the ISM system is IBM DB2 Universal Database - Extended Enterprise Edition. At a high level, ISM is a web based application system that's focal point is a database containing job postings, skills, applicants, and employers. The Java components that drive the web page creation get much of their content from the DBMS. In order to make the DBMS access efficient, consistent and modular, the majority of database access will be performed through stored procedures. The use of stored procedures also provides a means to enforce business logic rules. In the case of very simple selects from the database, queries are used directly instead of stored procedures.

## Connecting to the Database

A database connection is made initially at the time of login to validate the user and establish his identity as a job seeker, employer, or ISM staff. After establishing that, a virtual connection is made to the database using a generic username that corresponds to the user type. NAS holds a pool of sessions open to the database running under these generic usernames and matches the request for a connection to one of these.

## Stored Procedure Creation

Stored procedures in DB2 are written in Java. DB2 provides a tool to generate a suitable Java wrapper around the actual stored procedure SQL code. DB2 Class libraries are provided to gain access to the DBMS.

## Calling Stored Procedures from Servlets and JSPs

Stored procedures are called from within Servlets and JSPs using the Java Database Connectivity (JDBC) API. NAS supplies the engine that accepts these JDBC calls and passes them to the database server.

## Partitioned Database Cluster Issues

ISM is making use of a multi-server database cluster. DB2 allows for the partitioning of data in a single database amongst multiple servers. The ISM database is partitioned in such a way that most employer data resides on one server, and job seeker data resides on the other server. This improves performance by facilitating parallel database activities.



## Validating Input Data

Typically, after a user is done entering data onto a web page form, some button click or other control function is performed by the user. At this point, validation of data is performed. The ISM system performs validation and enforcement of business logic at three different levels:

1. On the input form web page itself
2. At the application server
3. At the database

### Validation on the Web Page

Very little validation will be done on the web page itself. The only validation done here is to make very simple checks for data type and format correctness and required fields. Doing this type of simple validation at the web page level improves performance by avoiding going to the application server with data that is obviously incomplete or incorrect. These validation checks are implemented with JavaScript. Form submission to the Servlet is blocked until these validations are passed.

### Validation at the Application Server

Entered data is passed to an Servlet when the user performs some action that causes the page to be submitted. At this point, the Servlet is responsible for a complete validation of all data as well as any other processing. Validations that were done at the web page are repeated in the Servlet or JSP to guard against malformed or insidious transmissions. All individual data items and referential integrity checks should be performed by the Servlet prior to submitting any data update requests to the database.

### Validation at the Database Server

The database server stored procedures, triggers, referential integrity, and key constraints enforce database integrity. However, by the time a simple data add or update request is made to the database by the Servlet, it is expected that the Servlet has thoroughly verified the integrity and should be without errors. In these cases, logic in the stored procedure need only return success or failure to the Servlet. In other cases, the logic of the stored procedure is more elaborate and needs to return more detailed status information.

## Error Handling

When errors occur in the application, they may need to be logged and/or reported to the user. If they are severe in nature, an administrator or other system support personnel must be notified. An infrastructure has been built into the ISM system to provide a central error delivery mechanism. It provides the ability to report simple errors, such as displaying messages on the web page form, as well as escalating errors to the point of sending Email, delivering SNMP traps, and sending alphanumeric pages.

Errors to be reported fall into several categories. The delivery of appropriate error messages to the appropriate destinations must be initiated by the module that traps the error. This could occur on the web page, in the application server, or at the database server.

### Displaying Errors on the Web Page

Errors are tracked and stored using cookies on the client computer. Whenever an error message needs to be displayed, the error is recorded in a cookie. When the page is rendered, the cookie(s) is/are read by another JavaScript module and displayed on the page. The key to this mechanism working, is forcing the page to be re-rendered from various points of execution.

Some simple validations are performed on the client computer using JavaScript embedded on the web

page. Standard error message handling JavaScript routines are available on any page to record the error. Once errors are recorded, they can be displayed by making appropriate function calls.

When errors are detected by a Servlet on the application server, the ISM error handler is called. Messages are delivered back to the client computer by sending cookie(s) in the response header, along with a request to go back to the previous page.

### Severe Error Logging and Reporting

When severe errors occur, the NAS error handling extension is used to handle them. Again, these errors could be triggered either at the browser, the application server, or the database server.

When errors occur that are severe in nature on the web page, a JavaScript function is available to invoke the error handling Servlet on the NAS server for severe error notification. This JavaScript function is part of the standard master templates. If a severe error should occur in the application server's code execution, the methods available from the error handler are called directly.

Some database severe errors are returned to the Servlet and reported on from there. Others that may occur independent of any specific request are reported through normal DB2 alerts provided for within the DB2 infrastructure.

### Security Handling

There are several elements in the ISM system that will have limited access. This includes seeing certain screen elements on pages, seeing entire pages, and performing certain actions on pages. To facilitate these requirements, a security infrastructure has been put into place via a NAS extension. This extension provides an easy interface to security information about the current user and what his permission levels are to perform actions or see items.

#### Users and Groups

A simple user/group/permission arrangement is used in the ISM system. Permissions are granted; there is no facility to specifically deny access to something. Permissions items are defined for anything that needs protection. Any group or user may be granted a permission. Users may belong to more than one group. Groups may belong to one or more groups. If the specific user, or any group he belongs to has the permission granted, then access is allowed.

#### Checking Access Rights

Permission items are defined and checked for in the JSP and Servlet to decide if the user should be shown items in the template, or allowed to access functions. Another technique used from the JSP or Servlet is to choose between various web pages based on user permissions. Raw navigational issues are handled by the base object infrastructure. Permissions are maintained that list valid from/to combinations of pages and page permissions.

#### Database Users

The DBMS has its own security infrastructure. To provide an extra layer of protection to the data, several different users are defined with varying access rights to the data itself. At the time the database connection is made, the user is identified and mapped to an appropriate database user according to the level of access allowed.

## Online Application Organization

This section addresses the specific sections of the online system. Application logic elements and implementation approaches are discussed.

The ISM system can be broken down into four sections:

1. Job seeker functions
2. Employer functions
3. ISM Staff functions
4. Administration functions

The sections that follow deal with each of these functional subsystems and any specific design issues and approaches taken within them. A general description of application flow is also provided. These following sections address major functional areas of the system. It is not intended to be an exhaustive inventory of all screens and functions.

### Job Seeker Functions

#### Registration

A job seeker begins his experience with the ISM system by registering. Only registered users with a username and password can use the system. Registration is a simple sequence of forms. After registration is completed, the user can logon to the system.

#### Skills Profile Entry

A series of forms is available to walk through the predefined skills and add them to the user's profile. The user chooses skills and assigns proficiency or experience levels to them. Extensive searching is available to choose skills related to various job titles. This functionality is also available in the employer section to define the required skills for a job order.

#### Skill Matching

Once a job seeker has filled in his skills profile, the skills matching function can be performed. This is the heart of the ISM system. Available job orders are compared with the user and a list of matching job opportunities is presented. Links to the detailed job information is then available. A link to MapQuest(r) is also provided for driving directions.

### Employer Functions

#### Registration

Employers must be registered prior to posting any job orders into the system. Once the employer goes through the online registration, job order worksheets can be prepared. However, these job orders cannot be posted to the system until an IDES staff member has reviewed the registration to validate the legitimacy of the entity.

#### Job Order Functions

Job orders are the key element for employers. Job openings are described in detail and entered into the system. Skills and proficiency/experience levels are assigned to the job order. After a job order is completed, a trial match can be performed. This function allows the employer to get a feel for how many qualified candidates exist in the ISM database. Modifications can then be performed prior to the actual

posting of the job order. Once posted, a match is performed and a list of qualified candidates is generated in the database. The next time a qualified candidate logs onto the system, that new job will appear in their list of qualified jobs.

### **Qualified Candidates**

Once qualified candidates have been identified through the matching process, the employer can perform actions to view the job seeker information and make referrals.

### **Referral Actions**

The referral actions are what triggers notifying the job seeker of a match in skills between them and a job order. Notifications are queued and processed in batch mode. Possible notification methods are an email, automated phone notification, or a letter.

### **Staff Functions**

The staff menu is presented when a IDES staff user logs on. The staff menu contains links to every function available to the job seeker and the employer. Additional functions available to staff are described in the sections below.

### **Employer Registration Requests**

Employers can be registered by IDES staff, or employers can submit their own registration requests. Once the request has been made, IDES staff review the validity of the company and then make the employer's registration active.

### **Search Employer Info**

Search screens allow the staff member to look up company information and edit their contact and any other information as needed. Employer information can also be printed.

### **Job Order Search**

A job order search screen provides staff members with a method to search for and edit a specific job order. Job order information can also be printed.

### **Administration Functions**

Administrative screens are used to maintain the various basic data of the system such as skills definitions, staff users, security settings, and other table maintenance.

## Section 4

# Batch Components

## Batch Components

This section presents the structure of the batch portion of the ISM application. Batch programs implement interfaces between the ISM system and legacy systems. Batch programs are also used to satisfy regular report generation requirements. General topics in this section include Unix batch execution environment, program elements, checkpoint/restart, and data transfer between the ISM Unix environment and the CMS IBM mainframe environment. Some specific batch program implementation details are also presented.

## General Mechanics and Approaches

This section presents general topics and approaches to implementing the batch system for ISM.

### Execution Environment

The batch programs for the ISM system are executed on one of the database servers. The second database server is available as a backup in the event that the primary batch processing database server is unavailable. Batch job execution is controlled by OSM COSbatch batch scheduling software. Batch jobs are registered in COSbatch for processing either at specific times, or on some other event, such as the existence of a data file or successful completion of another batch job.

### Program Elements

#### Shell Scripts

The batch programs for the ISM system run in a Unix environment. In Unix, batch jobs can be either a single executable program, or a shell script. A shell is simply a term for the command line interface to the operating system. Several different shell programs are available in the Unix environment. Examples of these are Bourne, Korn, C, and Bash. ISM Batch jobs are written as Korn shell scripts. Korn shell is the most common and popular Unix shell. Within the Korn shell script, individual programs can be executed, environment variables can be used, and basic control structure constructs are available. Return codes from programs can be checked within the script. Return codes from the script can be checked by COSbatch.

#### COBOL Programs

The core processing of the batch programs will be written in Microfocus COBOL. COBOL is the best suited tool for database access and file processing to and from the mainframe.

### Checkpoint/Restart

Two of the most important design features of ISM batch jobs is their ability to be restartable and their use of checkpoints. Many of the programs in ISM will be dealing with large amounts of data. If for some reason the job is interrupted, the ability to restart the job and have it resume processing where it left off saves valuable processing time and reduces performance impact on the system. From an operational standpoint, this approach offers simplicity. Any program can be terminated and restarted without the need for a lengthy manual rollback process.

#### The Batch Control Table

Central to the checkpoint/restart infrastructure is a batch control table that contains key information about the execution parameters and status of the job. Information contained here includes input/output file name(s), the current status of the job, and an indicator of where in the file the last checkpoint



occurred. There is also a checkpoint governor stored in the table that indicates the number of records to be processed in between checkpoints. This allows for some tuning of resource utilization. This technique limits the number of database locks and the length of time that records stay locked. The checkpoint value is read at the end of each checkpoint interval so that the parameter can be set dynamically.

## File Transfer

Many batch programs in the ISM system either generate a data file for the mainframe from data contained in ISM, or read a file created on the mainframe and post the information into the ISM database.

The mechanics of sending and receiving files between the ISM batch server system and the IBM mainframe will consist of dropping files off and reading them from a specific location on the IDES network. The specific mechanism of transfer and location have not yet been defined. Most likely, the transfer mechanism will be FTP or an NFS mounted volume that can be accessed directly. The intention is to avoid manual intervention in all file transfers for ISM. Files should be dropped off and picked up by the programs automatically with no human intervention.

## **Section 5**

# **Infrastructure Components**

# Infrastructure Components

This section presents the infrastructure upon which the ISM system is built. The ISM infrastructure can be defined as those components that provide core services to the rest of the application components. Much of the infrastructure centers around Netscape Application Server. A discussion of NAS architecture, as well as other components and how they interact is discussed. The use of application development languages is also addressed.

## NAS Architecture Overview

This section provides an overview of the components and features of the server and application architecture of the Netscape Application Server (NAS). The purpose is to provide basic background information to aid in understanding the ISM Application framework and the context in which it operates.

### Background

This section assumes that the reader has a basic background in general Internet, web, and web application technologies. If this is not the case then the reader may wish to read the "Technology Background" section.

### Introduction

The Netscape Application Server (NAS) is an application server product currently developed and marketed by Netscape Communications Corporation.

NAS offers a stable and scalable environment on which to develop and deploy robust and complex transaction based web applications.

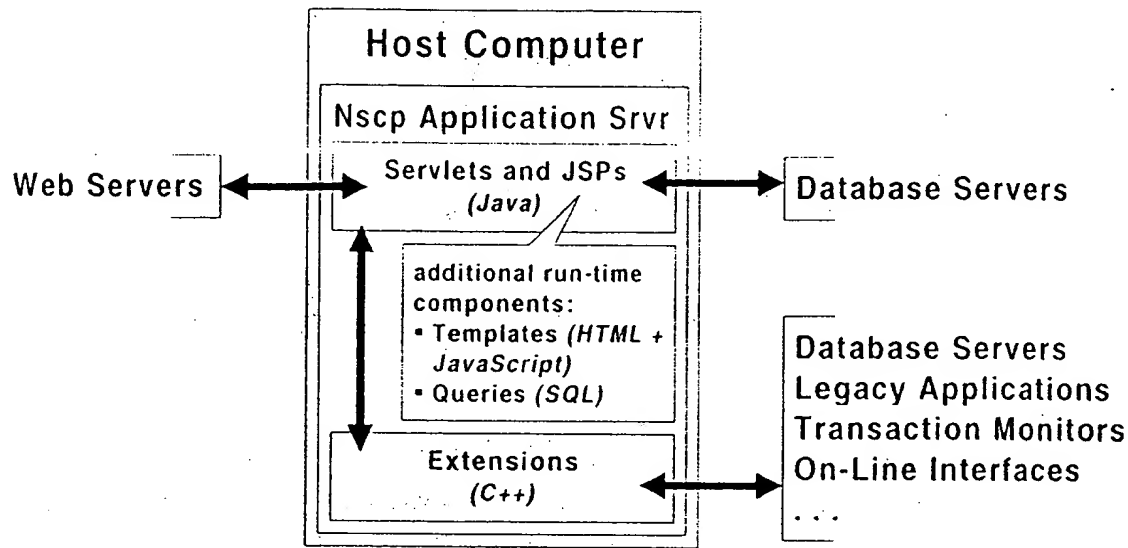
### Components

NAS based applications consist of off-the-shelf NAS servers to provide the core services and custom built application components to implement the application's specific business logic requirements. The custom built application logic components that execute on the server side consist of Java Servlets, Java Server Pages (Java imbedded in an HTML document), and application server extensions written in Java and C++.

As requests are received from the web user, via the web server, a specific Servlet is invoked to handle that request. The Servlet can access external resources such as databases. After processing is completed, a Servlet will typically either respond with an HTML stream back to the client, dispatch control to a Java Server Page (JSP), or a combination of the two. The Servlet or JSP can also use the services provided by the Extensions. The NAS Extensions function much like assembler exit routines on main frame applications. These extensions extend the core capabilities of the base NAS product to provide such functionality as persistent connections to back-end legacy applications, integration with transaction monitors, integration with third party packages, etc.

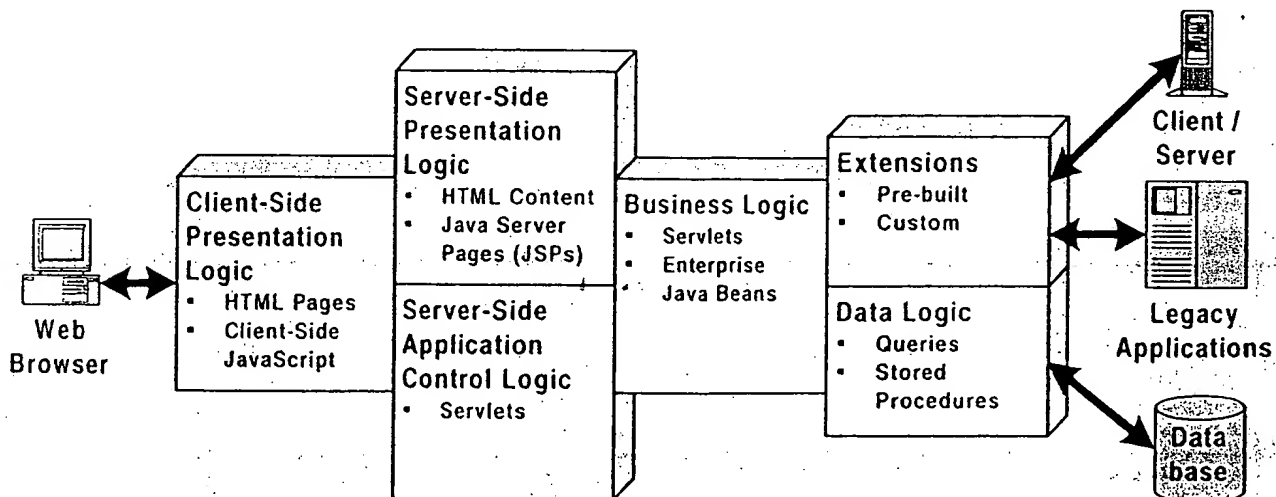
Figure 5-1 illustrates at a high-level how the Servlets, JSPs and Extensions work within a NAS server and the points of interaction with the web server, database servers, and other external services.

Figure 5-1 NAS Server Components



Structuring the NAS application architecture to use separate components for static pages, dynamic page templates, query files, and executable logic provides a multi-tier application model. A great deal of flexibility is available in matching the best module type to the application module's task. The advantages of this scheme are that the application components are separated into manageable pieces according to the skills required to prepare them and by the functions that they perform. This also allows for greater re-use of components, simpler testing, and modular deployment. This supports a higher quality development result and minimizes the impact on system availability when deploying ISM application software upgrades. Figure 5-2 illustrates the tiers of a web application and which NAS and other components address which tier.

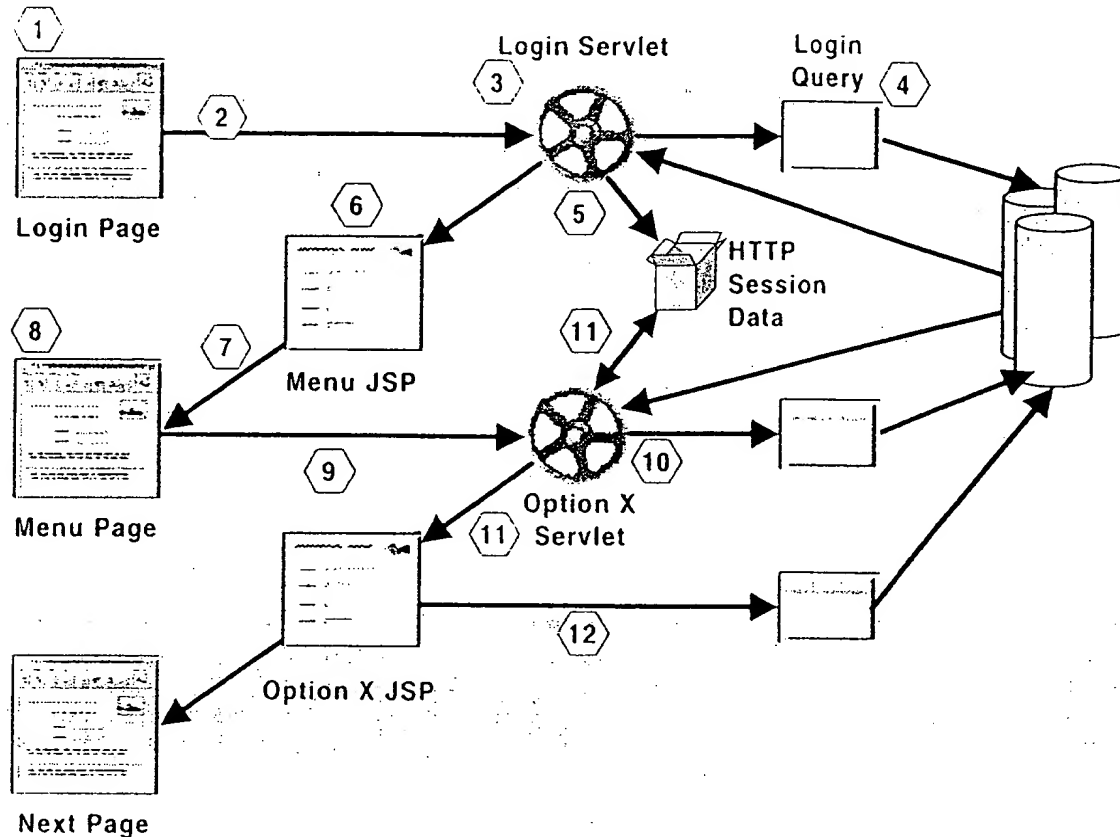
Figure 5-2 - Web Application Tiers



## Request Flow

Figure 5-3 flow of a NAS based application.

Figure 5-3 - NAS Application Flow



1. Within a web browser, a user is viewing the "Login" page containing a data entry form. The user enters their user name and password and clicks on the "Login" button.
2. The request, containing the values entered onto the web form, is sent through the web server to the application server.
3. The application server receives the request and runs the "Login" Servlet.
4. The Servlet retrieves the user's user name and password from the incoming parameters and uses the "Login" query to perform a search within the database to verify those credentials and to retrieve information about this user.
5. Once the credentials have been verified, the Servlet generates a new session identifier and creates a new container (HTTP session object) to hold information pertaining to this user such as the user's user name.
6. The Servlet then dispatches to the Menu JSP to generate a menu page customized for that user.
7. As the resulting page is created it is sent back to the web browser via the web server. Note that the new session identifier is also sent to the web browser via an HTTP cookie.
8. The "Menu" page is received and rendered by the browser. The user can then click on any of the options (links and forms) on that page.
9. When the user clicks on an option a new request is sent through the web server to the application server. Note that the web browser also sends the session identifier via an HTTP cookie.
10. The application server receives the request and runs the appropriate Servlet.

11. The Servlet retrieves all of the incoming parameters, including the session identifier. The Servlet can then use that session identifier to access the existing HTTP session "object" for that user and modify the information contained within it. The Servlet performs any necessary data access and dispatches to the appropriate JSP to prepare the next page for the user.
12. Optionally, the JSP can make necessary calls to database to retrieve additional data.

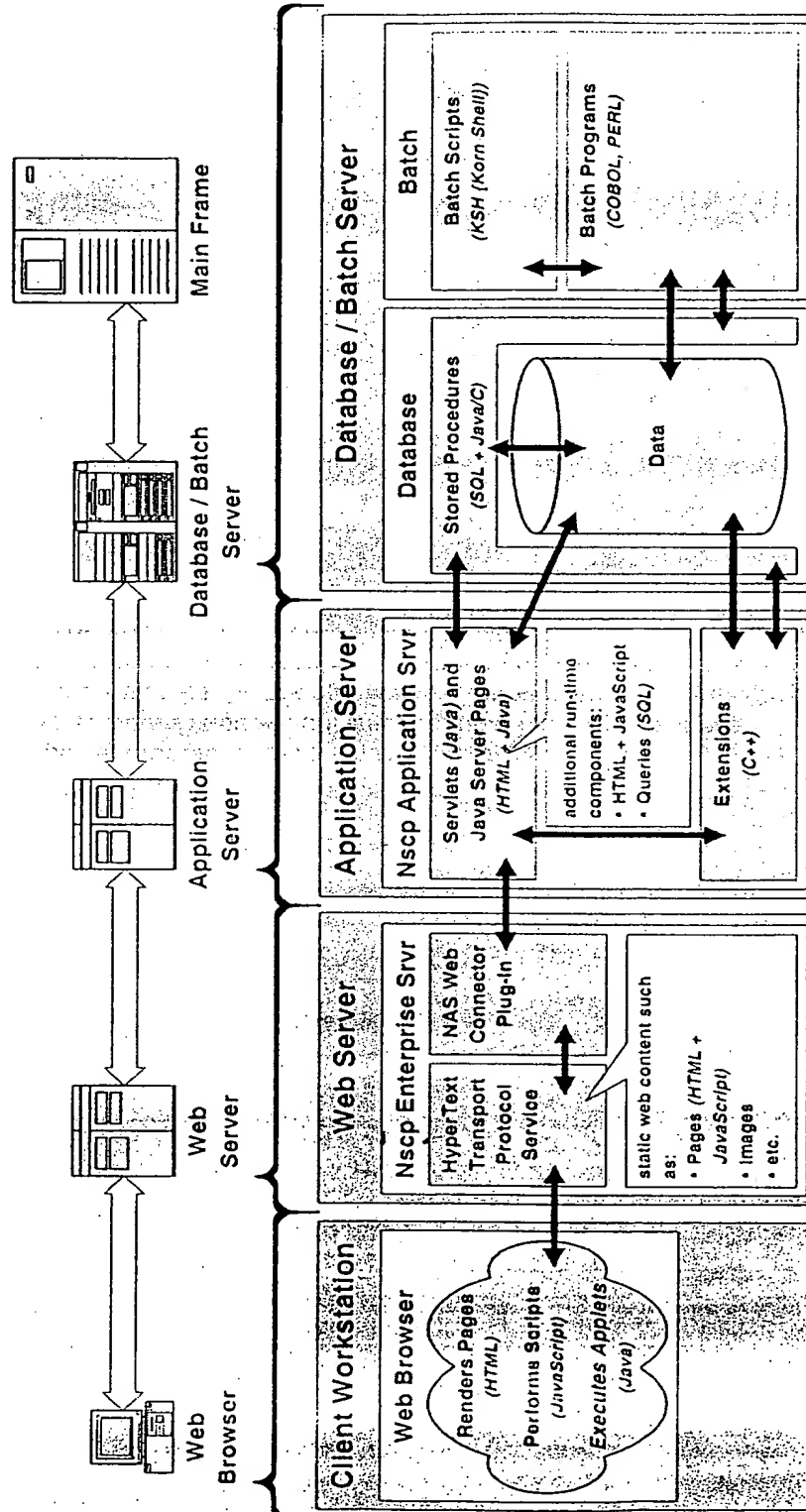
### Product Suite

The NAS product offerings include...

- Netscape Application Server (NAS) - a full-featured application server offering high performance, a high degree of fault tolerance and failure recovery, sophisticated session tracking, and dynamic statistics-based load balancing.
- Netscape Application Builder (NAB) - a graphical integrated development environment (IDE) for developing NAS application components for responding to page requests. These components include Java Servlets, SQL query modules, and JSPs. NAB facilitates building robust and fault tolerant applications to be deployed to a cluster of NAS servers.
- Netscape Extension Builder (NEB) - a graphical integrated development environment (IDE) for developing and deploying NAS extensions for providing additional core services to a NAS server.

## Component Overview Diagram

The ISM system consists of several components including host computers, operating systems, off-the-shelf application software, and custom designed software. The Component Overview Diagram illustrates these components.



## Component Definition

### Platforms

The ISM system employs multiple physical tiers: web client, web server, application server, database server, and database server. Each of these platforms provides specific services.

### Web Client (web browser)

The web client functions as the end-user interface for employer, applicant, and staff users. The web client presents a screen (web page) to the user and allows the user to interact with that screen - entering and changing data and activating controls such as submission buttons. The types of content to be handled by the web browser will be HTML documents with embedded images and JavaScript. The JavaScript provides for dynamic interaction with the web page within the web browser. The use of a web browser provides an open standards based interface and communication mechanism for interacting with the ISM system - those standards being HTML, JavaScript, and HTTP. The use of a web browser and also eliminates the need for maintenance and distribution of specialized application client software.

### Internet

The communications network connecting the clients to the servers will be the IDES private intranet and the public Internet. The use of the public Internet assures broad accessibility of the ISM system by the potential employer and applicant users as well as IETC partner staff users.

### Web Server

The web server receives and responds to all requests from the web clients. Requests for static (fixed, not dynamically generated) content are handled by the web server directly. Requests for dynamically generated screens - screens containing information residing in the ISM database - will be forwarded to the application server. The use of a web server provides an open standards based communication mechanism to the ISM system from the client systems - that standard being the HTTP protocol.

### Application Server

The application server consists of off-the-shelf application server software and custom built application components. The application server provides the basic services for a high-volume, fault-tolerant, transaction based application and provides the environment in which the custom built application components run. The application components consist of application logic written in Java (Java Servlets), output templates written in HTML and Java (Java Server Pages) and possibly containing embedded JavaScript, query definitions written in SQL, and application server extensions written in C++.

### Database Server

The database server will consist of off-the-shelf data base server software and custom built stored procedures. The stored procedures will be written in SQL for the data manipulation and either Java, C, or COBOL for the application logic and flow scripting around the SQL.

### Batch Server

The batch server will be co-located on a single computer platform with the database server. The batch server will execute batch business processing - not directly interacting with an end user. The batch processing will be written using ksh (Korn Shell command language scripting), COBOL, and PERL.



## Programming Languages Used for ISM

The custom built application components of the ISM system will be built using several different languages. These languages are all based on open standards and are supported by a large talent pool within the computer industry at large in the U.S.A. Each language has specific strengths and shortcomings and are chosen for specific uses based on those features and based on the support for those languages within the different platform components of the ISM system. The remainder of this section describes these programming languages.

A table outlining what these components will be used for and where they will reside and execute follows this section.

### Programming Language Definitions

#### Java

Java is an open standards based language created and overseen by Sun Microsystems. According to Sun, Java is a "simple, object-oriented, distributed, interpreted, robust, secure, architecture neutral, portable, high performance, multithreaded, and dynamic language." The Java language has been ported to every major computer platform including Sun Solaris, Microsoft Windows 3.x/95/98/NT, Macintosh OS, and every major UNIX implementation. One of the design goals of the Java language is that programs written in Java on one platform will run correctly on all other platforms that support Java without re-writing or re-compiling that program. Normally, the goals of portability and speed of execution are mutually exclusive: achieving speed is usually accomplished by compiling the program to low level instructions directly executable by the computer's processor - this makes the resulting code run very fast but can not be run on any dissimilar platform; achieving portability is usually achieved by leaving the code in a platform neutral format which must be interpreted on each computer where it is executed. Java accomplishes its goals of being portable and fast by using a hybrid approach: the programs are compiled to byte codes which are interpreted by a special program called the Java Virtual Machine (JVM) which, in turn, executes the corresponding native instructions of the host computer. On most computer platforms, the Java byte codes have a direct correlation to native instructions and, therefore, the JVM can execute the instructions quickly.

Java can be used to build many types of programs including...

- Stand-alone application including an interactive graphical applications as well as back-end batch and network server programs
- The write-once-run-anywhere nature of Java makes it particularly useful on the Internet where many different computer and operating system types are used as platforms for web browser applications. A single Java application can be delivered to, and executed on, web browsers running on any of a large number of computer platforms. These web delivered Java programs which run within a web browser are called Applets. Compiled Java programs can be quite large and slow to retrieve over a slow Internet connection.
- Several other software manufacturers have added the ability to write custom embedded functions using the Java language, for example: DB2 UDB supports writing stored procedures in Java with embedded SQL; Oracle Application Server supports writing custom functionality using Java via their J/Web cartridge.
- The Netscape Application Server application architecture supports developing on-line application logic using Java.

#### JavaScript

JavaScript is a lightweight interpreted programming language with object-oriented capabilities. JavaScript is an open standards based language created by Netscape and controlled by the European Computer Manufacturers Association (ECMA), a European association for standardizing information and

communication systems. The general-purpose core of the language has been embedded in Netscape Navigator, Microsoft Internet Explorer, and other web browsers.

JavaScript allows executable content to be included in web pages. Embedded JavaScript can be used to control document appearance and content, control the browser, interact with HTML Forms, interact with the user, etc. For example, JavaScript can be used to perform validation of input fields on an HTML form before submitting the request to the web server. JavaScript scripts can also perform dynamic control of the web page within the web browser to provide functionality such as moving and layering page elements to create a tab screen effect and showing and hiding form controls to create a disabled/enabled effect.

JavaScript programs (scripts) are plain text and are interpreted (not compiled). As a result, these scripts execute slower but the JavaScript files are tend to be small and easy to transmit to web browsers accessing the Internet through slow connections. An alternate to using JavaScript is to use another scripting language such as Microsoft's VBScript based on Microsoft's Visual Basic. A significant limitation of VBScript is that it can run only within Microsoft web browsers and it is not an open standard approved by any standards organization. Another choice is to use Java Applets. Java is an open standard and Java applets can run on most current web browsers but Applets tend to be much larger and, therefore, take longer to transmit over slow Internet connections and, therefore, should be used sparingly.

## HTML

HTML is an open standards based language for platform independent World Wide Web page layout description. The HTML standard is controlled by sub groups of the World Wide Web Consortium (W3C). HTML files are text based and tend to be small and easy to load over a slow connection to the Internet. The HTML standard continues to evolve with new features being added continually. As these new features are implemented and supported by the web browser manufactures then content providers use those new features in their web pages. There is no other competing language for this purpose.

## SQL

Structured Query Language (SQL) is an open standards based language for Data Definition and Data Manipulation within relation database servers. SQL is supported by every major relational database vendor.

## C/C++

C is a general-purpose programming language which features economy of expression, modern control flow and data structures, and a rich set of operators. C is not a "very high level" language, nor a "big" one, and is not specialized to any particular area of application. But its absence of restrictions and its generality make it more convenient and effective for many tasks than some higher level languages. It has been closely associated with the UNIX system where it was developed, since both the system and most of the programs that run on it are written in C. The language, however, is not tied to any one operating system or machine; and although it has been called a "system programming language" because it is useful for writing compilers and operating systems, it has been used equally well to write major programs in many different domains. C is a relatively "low level" language in that it deals with the same sort of objects that most computers do, namely characters, numbers, and memory addresses making it more akin to assembler languages than to a higher level language such as COBOL. Due to the low level of the programming features supported by C/C++ these programs tend to be difficult to program, difficult to debug, and bugs can have more severe consequences. C also provides the fundamental control-flow constructs required for well structured programs. C++ is an object oriented enhancement of the C programming language. Both C and C++ were created by AT&T and have ANSI standards. Programs written in C/C++ are compiled to binary executable code capable of executing on

only the computer platform for which it was compiled. Due to the lower level of the programming and the fact that C/C++ is compiled, programs written in C++ have among the fastest execution times of any other programming language - other than assembler.

## COBOL

COBOL (COmmon Business Oriented Language) is one of the most widely used programming languages for business applications in the world. It is particularly well suited to record processing and financial business processing. COBOL has an ANSI standard. COBOL provides a higher level of programming than does C/C++ and does not allow the developer to perform direct manipulation of the underlying computer platform. As such, these programs tend to not exhibit the same bug severity as do programs written in a lower level language but also are a poor choice for doing system service level development.

## PERL

PERL (Practical Extraction and Reporting Language) is a commercially-supported cross-platform general purpose scripting language invented in 1987 by Larry Wall. Perl has become the language of choice for World Wide Web development, text processing, Internet services, mail filtering, systems administration, and many other tasks requiring portable and easily-developed solutions. It is commonly used for job control scripting to function as the "glue" for connecting applications that normally would not talk to one another. Perl programs are interpreted and, therefore, can run on many different platforms without modification. Perl is also secure, object-oriented, robust, easy to learn and use, concise, and flexible.

## KSH

The Korn Shell is one of the standard command interfaces for UNIX systems. The Korn Shell also allows for the execution of scripts built using the Korn Shell command language features. Such scripts can be full fledged programs but are often used as job control "wrappers" performing job setup, invoking executable programs, and performing job clean-up much like JCL are used on the main frame.

**Component / Language / Platform Matrix**

The following table details the various languages used, the type of application components built using them, when these components are deployed and stored, and where these components eventually are executed in the course of handling a client transaction.

Language	ISM Purpose	Where Stored	Where Executed
Java	On-line, transaction based application logic implemented as Java Servlets and Java Server Pages within the application server.	application server	application server
	Applets: Complex graphical presentation and user interaction within web pages. Due to the larger size of applets and the longer loading time over slow Internet connections, applets will be used only on an as-needed basis.	web server	web browser
* - only one of Java, C, COBOL will be used for stored procedures.	Scripting / Logic processing within a DB2 Stored Procedure	database server	database server
JavaScript	Simple logic processing with a web page such as... 1) dynamic page manipulation such as showing and hiding content and form controls 2) verifying user data entry for completeness and validity before submitting the request	1) in separate JavaScript files and embedded in static HTML pages stored on the web server 2) embedded in output Java Server Pages stored on the application server	web browser
HTML	Static page - web page content and layout. May contain embedded JavaScript.	web server	web browser
	Java Server Pages used by the application server to prepare a web page in response to a request. The JSP may contain anything that is intended to be sent to the browser, including embedded JavaScript.	application server	1) used by application logic to generate dynamic pages 2) these generated pages are delivered to the web browser and rendered
SQL	Used directly by the application logic components within the application server to perform direct data queries against the database server.	application server	database server

Language	ISM Purpose	Where Stored	Where Executed
	Used directly by the batch logic components to perform direct data queries against the database server	batch server	database server
	Used within DB2 stored procedures to specify the data manipulation.	database server	database server
C++	Used to build NAS Extensions to extend the core functionality of the NAS server. Services to be provided include 1) error handling and notification 2) statistics gathering and monitoring 3) security rule enforcement	application server	application server
* - only one of Java, C, COBOL will be used for stored procedures.	Scripting / Logic processing within a DB2 Stored Procedure	database server	database server
COBOL	Batch programs executing business functions which access or manipulate the data within the ISM database. Examples include the various interfaces between the ISM System and main frame applications.	batch server	batch server
* - only one of Java, C, COBOL will be used for stored procedures.	Scripting / Logic processing within a DB2 Stored Procedure	database server	database server
PERL	Batch programs to perform various special purpose system support type operations such as parsing and reformatting files. This will be used on a limited basis.	batch server	batch server
KSH	KSH scripts will be used to control the execution of the COBOL and PERL batch programs much like JCL is used on the main frame. These scripts will perform operations such as job setup (such as copying and renaming files), program execution, return code status checking, and job clean-up.	batch server	batch server

\* DB2 UDB supports the use of Java, C/C++, and COBOL for developing stored procedures. Only one of those languages will be selected but that decision has not yet been made.